

A GRASS GIS parallel module for radio-propagation predictions

Lucas Benedičič, Felipe A. Cruz, Tsuyoshi Hamada and Peter Korošec

Abstract—Geographical information systems are ideal candidates for the application of parallel programming techniques, mainly because they usually handle large data sets. To help us deal with complex calculations over such data sets, we investigated the performance constraints of a classic master-worker parallel paradigm over a message-passing communication model. To this end, we present a new approach that employs an external database in order to improve the calculation/communication overlap, thus reducing the idle times for the worker processes. The presented approach is implemented as part of a parallel radio-coverage prediction tool for the GRASS environment. The prediction calculation employs digital elevation models and land-usage data in order to analyze the radio coverage of a geographical area. We provide an extended analysis of the experimental results, which are based on real data from an LTE network currently deployed in Slovenia. Based on the results of the experiments, which were performed on a computer cluster, the new approach exhibits better scalability than the traditional master-worker approach. We successfully tackled real-world data sets, while greatly reducing the processing time and saturating the hardware utilization.

Index Terms—GRASS, GIS, parallel, radio, propagation, simulation.

I. INTRODUCTION

ALTHOUGH Gordon Moore's well-known and often cited prediction still holds [23], the fact is that for the past few years, CPU speeds have hardly been improving. Instead, the number of cores within a single CPU is increasing. This situation poses a challenge for software development in general and research in particular: a hardware upgrade will, most of the time, fail to double the serial execution speed of its predecessor. However, since this commodity hardware is present in practically all modern desktop computers, it creates an opportunity for the parallel exploitation of these computing resources to enhance the performance of complex algorithms over large data sets. The challenge is thus to deliver the computing power of multi-core systems in order to tackle a computationally time-consuming problem, the completion of which is unfeasible using traditional serial approaches. Moreover, by accessing many such computing nodes through a network connection, even more possibilities are available.

A traditional approach when dealing with computationally expensive problem solving is to simplify the models in order to

be able to execute their calculations within a feasible amount of time. Clearly, this method increases the introduced error level, which is not an option for a certain group of simulations, e.g., those dealing with disaster contingency planning and decision support [19], [46]. The conducted simulations during the planning phase of a radio network also belong to this group. Their results are the basis for the decision making prior to physically installing the base stations and antennas that will cover a certain geographical area. A greater deviation of these results increases the probability of making the wrong decisions at the time of the installation, which may considerably increase the costs or even cause mobile-network operators to incur losses.

Various groups have successfully deployed high-performance computing (HPC) systems and techniques to solve different problems dealing with spatial data [2], [3], [13], [19], [21], [27], [38], [39], [40], [44], [46]. This research has confirmed that a parallel paradigm such as master-worker, techniques like work pool (or task farming) and spatial-block partitioning are applicable when dealing with parallel implementations over large spatial data sets. However, it is well known that parallel programming and HPC often call for area experts in order to integrate these practices into a given environment [8]. Moreover, the wide range of options currently available creates even more barriers for general users wanting to benefit from HPC.

In this paper, we combine some of the known principles of HPC and introduce a new approach in order to improve the performance speed of a GIS module for radio-propagation predictions. The efficiency improvement is based on overlapping process execution and communication in order to minimize the idle time of the worker processes and thus improve the overall efficiency of the system. To this end, we save the intermediate calculation results into an external database (DB) instead of sending them back to the master process. We implement this approach as part of a parallel radio-prediction tool (PRATO) for the open-source Geographic Resources Analysis Support System (GRASS) [25]. For its architecture, we have focused on scalability, clean design and the openness of the tool, inspired by the GRASS GIS. This makes it an ideal candidate for demonstrating the benefits and drawbacks of several reviewed patterns, while tackling the radio-coverage predictions of big problem instances, e.g., real mobile networks containing thousands of transmitters over high-resolution terrains, and big-scale simulations covering the whole country.

Lucas Benedičič is with the Research and Development Department of Telekom Slovenije, d.d., Ljubljana, Slovenia, e-mail: lucas.benedicic@telekom.si. Corresponding author.

Felipe A. Cruz and Tsuyoshi Hamada are with the Nagasaki Advanced Computing Center, Nagasaki University, Nagasaki, Japan.

Peter Korošec is with the Computing Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia, e-mail: peter.korosec@ijs.si.

Preprint submitted to Taylor & Francis.

A. Objectives

In order to assess the benefits and drawbacks of various reviewed approaches from the performance point of view, we evaluate PRATO in a distributed computing environment. Furthermore, by presenting a detailed description of its design and implementation, we provide an analysis of the patterns achieving higher efficiency levels, so that they can be adopted for general task parallelization in the GRASS GIS.

The paper is organized as follows. Section II gives an overview of the relevant publications, describing how they relate to our work. Section III gives a description of the radio-coverage prediction problem, including the radio-propagation model. Section IV concentrates on the design principles and implementation details of the radio-propagation tool, for the serial and parallel versions. Section V discusses the experimental results and their analysis. Finally, Section VI draws some conclusions.

II. RELATED WORK

The task-parallelization problem within the GRASS environment has been addressed by several authors in a variety of studies. For example, in [6], the authors present a collection of GRASS modules for a watershed analysis. Their work concentrates on different ways of slicing raster maps to take advantage of a Message Passing Interface (MPI) implementation.

In the field of high-performance computing, the authors of [2] presented implementation examples of a GRASS raster module, used to process vegetation indexes for satellite images, for MPI and Ninf-G environments. The authors acknowledge a limitation in the performance of their MPI implementation for big processing jobs. The restriction appears due to the computing nodes being fixed to a specific spatial range, since the input data are equally distributed among worker processes, creating an obstacle for load balancing in heterogeneous environments.

Using a master-worker technique, the work by [18] abstracts the GRASS data types into its own *struct* and MPI data types, thus not requiring the GRASS in the worker nodes. The data are evenly distributed by row among the workers, with each one receiving an exclusive column extent to work on. The test cluster contains heterogeneous hardware configurations. The authors note that data-set size is bounded by the amount of memory on each of the nodes, since they allocate the memory for the whole map as part of the set-up stage, before starting the calculation. Regarding the data sets during the simulations, the largest one contains 3,265,110 points. They conclude that the data-set size should be large enough for the communication overhead to be hidden by the calculation time, so that the parallelization pays off.

In [38], the authors employ a master-worker approach, using one worker process per worker node. The complete exploitation of the computing resources of a single computing node is achieved with OpenMP. The experimental environment features one host. The horizon-composition algorithm presents no calculation dependency among the spatial blocks. Consequently, the digital elevation model (DEM) may be divided

into separate blocks to be independently calculated by each worker process. The authors present an improved algorithm that can also be used to accelerate other applications like visibility maps. The tasks are dynamically assigned to idle processes using a task-farming paradigm over the MPI.

Also, in [39] there is no calculation dependency among the spatial blocks. The experimental evaluation is made over multiple cores of one CPU and a GPU, communicated using a master-worker setup.

In [46], the authors present a parallel framework for GIS integration. Based on the principle of spatial dependency, they lower the calculation processing time by backing it with a knowledge database, delivering the heavy calculation load to the parallel back-end if a specific problem instance is not found in the database. There is an additional effort to achieve the presented goals, since the implementation of a fully functional GIS (or “thick GIS” as the authors call it) is required on both the desktop client and in the parallel environment.

An agent-based approach for simulating spatial interactions is presented in [11]. The authors’ approach decomposes the entire landscape into equally-sized regions, i.e., a spatial-block division as in [38], which are in turn processed by a different core of a multi-core CPU. This work uses multi-core CPUs instead of a computing cluster.

Some years ago, grid computing received the attention of the research community as a way of accessing the extra computational power needed for the spatial analysis of large data sets [3], [42], [43]. However, several obstacles are still preventing this technology from being more widely used. Namely, its adoption requires not only hardware and software compromises with respect to the involved parts, but also a behavioral change at the human level [3].

III. RADIO-COVERAGE PREDICTION FOR MOBILE NETWORKS

A. Background

The coverage planning of radio networks is a key problem that all mobile operators have to deal with. Moreover, it has proven to be a fundamental issue, not only in LTE networks, but also in other standards for mobile communications [31], [33], [36], [41]. One of the primary objectives of mobile-network planning is to efficiently use the allocated frequency band to ensure that some geographical area of interest can be satisfactorily reached with the base stations of the network. To this end, radio-coverage prediction tools are of great importance as they allow network engineers to test different network configurations before physically implementing the changes. Nevertheless, radio-coverage prediction is a complex task, mainly due to the several combinations of hardware and configuration parameters that have to be analyzed in the context of different environments. The complexity of the problem means that radio-coverage prediction is a computationally-intensive and time-consuming task, hence the importance of using fast and accurate tools (see Section IV-B for a complexity analysis of the algorithm). Additionally, since the number of deployed transmitters keeps growing with the adoption of modern standards [31], there is a clear need for a radio-propagation tool that is able to cope with larger work loads in

a feasible amount of time (see Section IV-B for the running time of the serial version).

In this work, we present PRATO: a high-performance radio-propagation prediction tool for GSM (2G), UMTS (3G) and LTE (4G) radio networks. It is implemented as a module of the GRASS GIS. It can be used for planning the different phases of a new radio-network installation, as well as a support tool for maintenance activities related to network troubleshooting in general and optimization in particular. Specifically, automatic radio-coverage optimization requires the evaluation of millions of radio-propagation predictions in order to find a good solution set, which is unfeasible using other serial implementations of academic or commercial tools [17], [22], [29].

As a reference implementation, we used the publicly available radio-coverage prediction tool, developed in [17]. The authors of this work developed a modular radio-coverage tool that performs separate calculations for radio-signal path loss and antenna radiation patterns, also taking into account different configuration parameters, such as antenna tilting, azimuth and height. The output result, saved as a raster map, is the maximum signal level over the target area, in which each point represents the received signal from the best serving transmitter. This work implements some well-known radio-propagation models, e.g., Okumura-Hata [15] and COST 231 [7]. The latter is explained in more detail in Section III-B. Regarding the accuracy of the predicted values, the authors [17] report comparable results to those of a state-of-the-art commercial tool. To ensure that our implementation is completely compliant with the previously mentioned reference, we have designed a comparison test that consists of running both tools with the same set of input parameters. The test results from PRATO and the reference implementation were identical in all the tested cases.

B. Propagation modeling

PRATO uses the COST-231 Walfisch-Ikegami radio-propagation model [33], which was introduced as an extension of the well-known COST Hata model [32]. The suitability of this model comes from the fact that it distinguishes between line-of-sight (LOS) and non-line-of-sight (NLOS) conditions.

In this work, as well as in the reference implementation [17], the terrain profile is used for the LOS determination. In this context, a NLOS situation appears when the first Fresnel zone is obscured by at least one obstacle [45]. We include a correction factor, based on the land usage (clutter data), for accurately predicting the signal-loss effects due to foliage, buildings and other fabricated structures. This technique is also adopted by other propagation models for radio networks, like the artificial neural networks macro-cell model developed in [24]. Consequently, we introduce an extra term for signal loss due to clutter (L_{CLUT}) to the Walfisch-Ikegami model [33], defining the path loss as

$$PL(d) = L_0(d) + L_{CLUT} + \begin{cases} PL_{LOS}(d) \\ PL_{NLOS}(d) \end{cases}, \quad (1)$$

where $L_0(d)$ is the attenuation in free space and is defined as

$$L_0(d) = 32.45 + 20 \log(d) + 20 \log(F). \quad (2)$$

If there is LOS between the transmitter antenna and the mobile, the path loss $PL_{LOS}(d)$ is defined as

$$PL_{LOS}(d) = 42.64 + 26 \log(d) + 20 \log(F), \quad (3)$$

whereas the path loss for NLOS conditions is determined as

$$PL_{NLOS}(d) = L_{RTS} + L_{MSD}. \quad (4)$$

Here, d is the distance (in kilometers) from the transmitter to the receiver point, F is the frequency (in MHz), L_{RTS} represents the diffraction from rooftop to the street, and L_{MSD} represents the diffraction loss due to multiple obstacles. Consequently, the total path loss from the antenna to the mobile device is calculated as in Equation (1), where the attenuation in free space and due to clutter are also taken into account.

IV. DESIGN AND IMPLEMENTATION

A. Design of the serial version

This section describes the different functions contained in the serial version of PRATO, which is implemented as a GRASS module. Their connections and data flow are depicted in Figure 1 on page 3, where the parallelograms of the flow diagram represent the input/output (I/O) operations.

Our design follows a similar internal organization as the radio-planning tool presented in [17], but with some important differences. First, the modular design was avoided in order to prevent the overhead of I/O operations between the components of a modular architecture. Second, our approach employs a direct connection to an external database server for intermediate result saving, instead of the slow, built-in GRASS database drivers. To explicitly avoid tight coupling with a specific database vendor, the generated output is formatted in plain text, which is then forwarded to the DB. Any further processing is achieved by issuing a query over the database tables that contain the partial results for each of the processed transmitters.

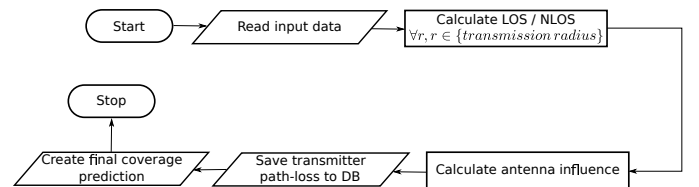


Figure 1. Flow diagram of the serial version.

1) *Isotropic path-loss calculation*: This step starts by calculating which receiver points, r , are within the specified transmission radius (see “*transmission radius*” in Figure 1 on page 3). The transmission radius is defined around each transmitter in order to limit the radio-propagation calculation to a reasonable distance. For these points, the LOS and NLOS conditions are calculated with respect to the transmitter (see “Calculate LOS/NLOS” in Figure 1 on page 3). The following

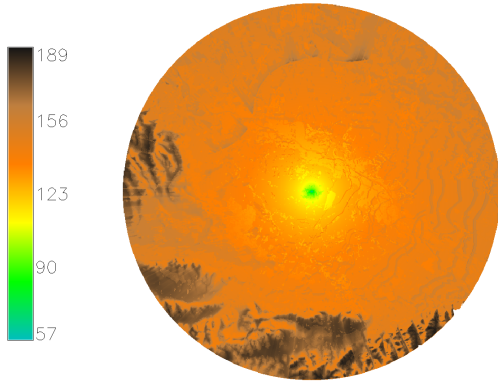


Figure 2. Example of raster map, showing the result of a path-loss calculation from an isotropic source.

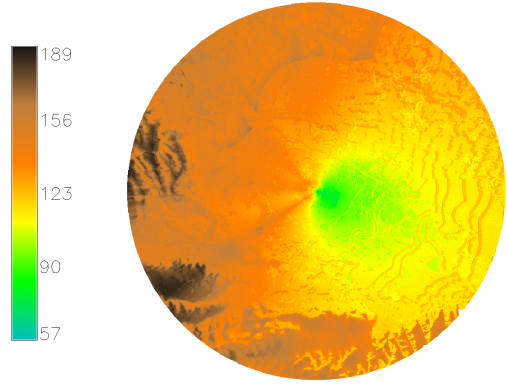


Figure 3. Example of raster map, showing the antenna influence over the isotropic path-loss result, as depicted in Figure 2 on page 4.

step consists of calculating the path loss for an isotropic source (or omni antenna). This calculation is performed by applying the Walfisch-Ikegami model, which was previously defined in Equation (1), to each of the points within the transmission radius around the transmitter.

Figure 2 on page 4 shows an example result of the isotropic path-loss calculation, only including the map area within the transmission radius. The color scale is given in dB, indicating the signal loss from the isotropic source of the transmitter, located at the center. Notice the hilly terrain is clearly distinguished due to LOS and NLOS conditions from the signal source.

2) *Antenna diagram influence*: This step considers the antenna radiation diagram of the current transmitter and its influence over the isotropic path-loss calculation (see “Calculate antenna influence” in Figure 1 on page 3). Working on the in-memory results generated by the previous step, the radiation diagram of the antenna is taken into account, including the beam direction, the electrical and the mechanical tilt. Figure 3 on page 4 shows the map area within the transmission radius, where this calculation step was applied to the results from Figure 2 on page 4. Notice the distortion of the signal propagation that the antenna has introduced.

3) *Transmitter path-loss prediction*: In this step, the path-loss prediction of the transmitter is saved in its own database table (see “Save transmitter path-loss to DB” in Figure 1 on page 3). This is accomplished by connecting the standard output of the developed module with the standard input of a database client. Naturally, the generated plain text should be understood by the DB itself.

4) *Coverage prediction*: The final radio-coverage prediction, containing the aggregation of the partial path-loss results of the involved transmitters, is created in this step (see “Create final coverage prediction” in Figure 1 on page 3). The received signal strength from each of the transmitters is calculated as the difference between its transmit power and the path loss for the receiver’s corresponding position. This is done by executing an SQL query over the tables containing the path-loss predictions of each of the processed transmitters. Finally, the output is generated, using the GRASS built-in modules *v.in.ascii* and *v.to.rast*, which create a raster map using the query results as the input. The final raster map contains the

maximum received signal strength for each individual point, as shown in Figure 4 on page 4. In this case, the color scale is given in dBm, indicating the strongest received signal strength from the transmitters.

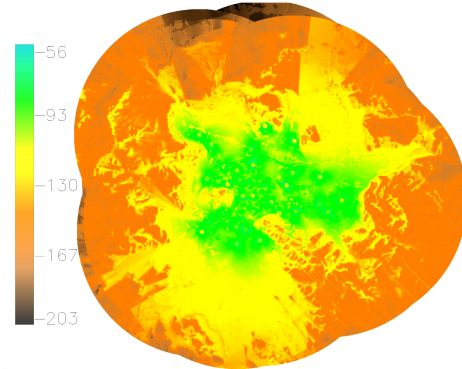


Figure 4. Example of a raster map, displaying the final coverage prediction of 136 transmitters over a geographical area. The color scale is given in dBm, indicating the received signal strength. Darker colors denote areas with a reduced signal due to the fading effect of the hilly terrain and clutter.

B. Computational complexity of the radio-coverage algorithm

Table I
PSEUDO CODE OF THE RADIO-COVERAGE PREDICTION ALGORITHM. THE TIME COMPLEXITY IS GIVEN PER LINE.

```

DEM ← Digital Elevation Model (DEM) of the whole area.      ▷ O(M)
Clutter ← signal Losses due to land usage of the whole area. ▷ O(M)
T ← transmitter configuration data.                          ▷ O(n)
for all t ∈ T do                                           ▷ O(n · m²)
    DEMt ← DEM area within transmission radius of t        ▷ O(m)
    Clutt ← Clutter area within transmission radius t       ▷ O(m)
    LoSt ← LineOfSight (DEMt)                             ▷ O(m²)
    PLt ← PathLoss (DEMt, Clutt, LoSt)                  ▷ O(m²)
    Diagt ← Antenna diagram of t                           ▷ O(1)
    PLt ← AntennaInfluence (Diagt, PLt)                  ▷ O(m)
end for
for all t ∈ T do                                           ▷ O(n · m)
    CoveragePrediction ← PathLossAggregation (t, PLt)      ▷ O(m)
end for
return CoveragePrediction

```

In this section, the time complexity of the radio-coverage prediction algorithm is presented, for which the pseudo code is listed in Table I.

The algorithm starts by loading the input, i.e., the DEM and the clutter data. Both regular square grids (RSGs) should account for the same area and resolution, consequently containing the same number of pixels, M . The transmitter data is then loaded into set T , the cardinality of which is denoted as $n = |T|$. For each transmitter $t \in T$, a smaller subarea of the DEM and clutter data (denoted DEM_t and $Clut_t$, respectively) is delimited around t , based on a given transmission radius. The number of pixels within this sub-area is denoted as m , and its value is the same for all $t \in T$. The visibility for an RSG cell is computed using the *LineOfSight* function, by walking from the antenna of the transmitter to the given element, along the elements intersected by a LOS, until either the visibility is blocked, or the target is reached [10]. Regarding the *PathLoss* function, whenever a receiver point is in NLOS, the walking path from the transmitter has to be inspected for obstacles, calculating the diffraction losses for each of them, i.e., L_{MSD} from Equation (4). Hence, its quadratic complexity, which dominates the complexity of the algorithm, together with *LineOfSight*, resulting in an algorithmic complexity denoted by

$$O(M + n \cdot m^2). \quad (5)$$

Although n will generally be many orders of magnitude smaller than m^2 , its computational-time complexity is relevant for practical use. For example, assuming the radio-coverage prediction for one transmitter completes in around 15 seconds using a serial implementation, the prediction for a mobile network comprising 10,240 transmitters would have an execution time of almost two days.

C. Multi-paradigm parallel programming

The implementation methodology adopted for PRATO follows a multi-paradigm, parallel programming approach in order to fully exploit the resources of each of the nodes in a computing cluster. This approach combines a master-worker paradigm with an external DB. To efficiently use a shared memory multi-processor on the worker side, and to effectively overlap the calculation and communication, PRATO uses POSIX threads [5].

To use the computing resources of a distributed memory system, such as a cluster of processors, PRATO uses the MPI [12]. The MPI is a message-passing standard that defines the syntax and semantics designed to function on a wide variety of parallel computers. The MPI enables multiple processes, running on different processors of a computer cluster, to communicate with each other. It was designed for high performance on both massively parallel machines and on workstation clusters.

In order to make the text clearer and to differentiate between the programming paradigms used from here on, we will refer to a POSIX thread simply as a ‘thread’ and a MPI process as a ‘process’.

D. Design of the parallel version

By maintaining our focus on the practical usability and performance of PRATO, we are introducing a parallel implementation to overcome the computational-time constraints that prevent a serial implementation of the radio-coverage prediction algorithm from tackling big problem instances in a feasible amount of time.

A major drawback of the GRASS as a parallelization environment is that it is not thread-safe, meaning that concurrent changes to the same data set have an undefined behavior [4]. One technique to overcome this problem is to abstract the spatial data from the GRASS. For example, in [18], the authors achieve the GRASS abstraction by introducing a *Point* structure with four *double* attributes, where each pixel of the RSG is mapped to an instance of this structure. Another possibility is for one of the processes, e.g., the master, to read entire rows or columns of data before dispatching them for processing to the workers [2], [18]. In this case, an independence between row/column calculations is required, which is a problem-specific property. In our case, we propose to achieve the GRASS abstraction by loading the spatial data into a 2D matrix (or matrices) of basic data-type elements, e.g., *float* or *double* depending on the desired accuracy. The geographical location of each element is calculated as the geographical location of the matrix plus the element offset within it. The advantage of this technique is having the geographical location of each pixel readily available with a minimum memory footprint. Moreover, a convenient consequence of this abstraction schema is that worker processes are completely independent of the GRASS, thus significantly simplifying the deployment of the parallel implementation over multiple computing hosts.

In the area of geographical information science, the master-worker paradigm has been successfully applied by several authors [1], [2], [6], [13], [18], [38], [39]. However, sometimes this technique presents certain issues that prevent the full exploitation of the available computing resources when deployed over several networked computers. Additionally, such issues are difficult to measure when the parallelization involves only one computing node [38], [39], i.e., no network communication is required, or only a few processes deployed over a handful of nodes [18]. Specifically, we are referring to network saturation and idle processes within the master-worker model. Generally speaking, a single communicating process, e.g., the master, is usually not able to saturate the network connection of a node. Using more than one MPI process per node might solve this problem, but possible rank-ordering problems may appear, thus restricting the full utilization of the network [30]. Another issue appears when the master process executes the MPI code, in which case other processes sleep, making a serial use of the communication component of the system. Consequently, the master process becomes the bottleneck of the parallel implementation as the number of worker processes it has to serve grows. This situation is also common when dealing with the metadata of a spatial region, which may relate to several elements of a RSG, making it a frequent cause of load imbalance [11], [16], [44]. In our case, the transmitter configuration and its antenna diagram

represent metadata that are complementary to the sub-region that a transmitter covers.

Hybrid MPI-OpenMP implementations [38], [39], in which no MPI calls are issued inside the OpenMP-parallel regions, also fail to saturate the network [30]. A possible solution to this problem is to improve the communication overlap among the processes. To this end, we have implemented non-blocking point-to-point MPI operations, and an independent thread in the worker process to save the intermediate results to a DB. We use one such database system per computer cluster, which also serves the input data to the GRASS, in order to aggregate the partial results of the path-loss predictions or to visualize them. It is important to note that any kind of database system may be used. By this we mean relational, distributed [28] or even those of the NoSQL type [37]. Nevertheless, in this study we use a central relational database system, since they are the most popular and widely available ones. Additionally, the non-blocking message-passing technique used to distribute the work-load among the nodes provides support for heterogeneous environments. As a result, computing nodes featuring more capable hardware receive more work than those with weaker configurations, thus ensuring a better utilization of the available computing resources despite hardware diversity and improved load balancing.

1) *Master process*: The master process, for which the flow diagram is given in Figure 5 on page 7, is the only component that runs within the GRASS environment. As soon as the master process starts, the input parameters are read. This step corresponds to “Read input data” in Figure 5 on page 7, and it is carried out in a similar way as in the serial version. The next step delivers the metadata that is common to all the transmitters and the whole region to all the processes (see “Metadata broadcasting” in Figure 5 on page 7). Before distributing the work among the worker processes, the master process proceeds to decompose the loaded raster data into 2D matrices of basic-data-type elements, e.g., *float* or *double*, before dispatching them to the multiple worker processes. In this case, the decomposition applies to the DEM and the clutter data only, but it could be applied to any point-based data set. In the next step, the master process starts an asynchronous message-driven processing loop (see “Processing loop” in Figure 5 on page 7), the main task of which is to assign and distribute the sub-region and configuration data of different transmitters among the idle worker processes.

The flow diagram shown in Figure 6 on page 7 illustrates the “Processing loop” step of the master process. In the processing loop, the master process starts by checking the available worker processes, which will calculate the radio-coverage prediction for the next transmitter. It is worth pointing out that this step also serves as a stopping condition for the processing loop itself (see “Any worker still on?” in Figure 6 on page 7). The active worker processes inform the master process that they are ready to compute by sending an idle message (see “Wait for idle worker” in Figure 6 on page 7). The master process then announces to the idle worker process that it is about to receive new data for the next calculation, and it dispatches the complete configuration of the transmitter to be processed (see “Send keep-alive message”

and “Send transmitter data” steps, respectively, in Figure 6 on page 7). This is only done in the case that there are transmitters for which the coverage prediction has yet to be calculated (see “Any transmitters left?” in Figure 6 on page 7). The processing loop of the master process continues to distribute the transmitter data among the worker processes, which asynchronously become idle as they finish the radio-prediction calculations they have been assigned by the master process. When there are no more transmitters left, all the worker processes announcing they are idle will receive a shutdown message from the master process, indicating to them that they should stop running (see “Send stop message” in Figure 6 on page 7). The master process will keep doing this until all the worker processes have finished (see “Any worker still on?” in Figure 6 on page 7), thus fulfilling the stopping condition for the processing loop.

Finally, the last step of the master process is devoted to creating the final output of the calculation, e.g., a raster map (see “Create final coverage prediction” in Figure 5 on page 7). The final coverage prediction of all the transmitters is an aggregation from the individual path-loss results created by each of the worker processes during the “Processing loop” phase in Figure 5 on page 7, which provides the source data for the final raster map. The aggregation of the individual transmitter path-loss results is accomplished by issuing an SQL query over the database tables containing the partial results, in a similar way as in the serial version.

2) *Worker processes*: An essential characteristic of the worker processes is that they are completely independent of the GRASS, i.e., they do not have to run within the GRASS environment nor use any of the GRASS libraries to work. This aspect significantly simplifies the deployment phase to run PRATO on a computer cluster, since no GRASS installation is needed on the computing nodes hosting the worker processes.

One possibility to overcome the thread-safety limitation of the GRASS is to save the transmitter path-loss predictions through the master process, thus avoiding concurrent access. However, for the workers to send intermediate results back to the master process, e.g., as in [1], [18], is a major bottleneck for the scalability of a parallel implementation. The scalability is limited by the master process, because it must serially process the received results in order to avoid inconsistencies due to concurrent access. Instead, our approach allows each of the worker processes to output its intermediate results into a DB, i.e., each path-loss prediction in its own table. Additionally, worker processes do this from an independent thread, which runs concurrently with the calculation of the next transmitter received from the master process. In this way, the overlap between the calculation and communication significantly hides the latency created by the result-dumping task, thus making better use of the available system resources.

The computations of the worker processes, for which the flow diagram is given in Figure 7 on page 8, begin by receiving metadata about the transmitters and the geographical area from the master process during the initialization time (see “Receive broadcasted metadata” in Figure 7 on page 8).

After the broadcasted metadata are received by all the worker processes, each one proceeds to inform the master

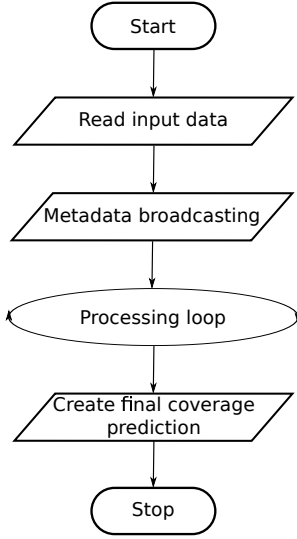


Figure 5. Flow diagram of the master process.

process that it is ready (i.e., in an idle state) to receive the transmitter-configuration data that defines which transmitter path-loss prediction to perform (see “Send idle message” in Figure 7 on page 8). If the master process does not give the instruction to stop processing (see “Has stop message arrived?” in Figure 7 on page 8), the worker process collects the sub-region spatial data and the transmitter configuration (see “Receive transmitter data” in Figure 7 on page 8). In the event that a stop message is received, the worker process will wait for any result-dumping thread to finish (see “Wait for result-dump thread” in Figure 7 on page 8) before shutting down. The coverage calculation itself follows a similar design as the serial version (see “Coverage calculation” in Figure 7 on page 8).

As mentioned before, the worker process launches an independent thread to save the path-loss prediction of the target transmitter to a database table (see “Threaded save path-loss to DB” in Figure 7 on page 8). It is important to note that there is no possibility of data inconsistency due to the saving task being executed inside a thread, since path-loss data from different workers belong to different transmitters and are, at this point of the process, mutually exclusive.

3) *Master-worker communication*: Similar to [38], [39], the message-passing technique used in this work enables a better use of the available computing resources, both in terms of scalability and load balancing, while introducing a negligible overhead. This last point is supported by the experimental results, introduced in Section V-C.

The first reason to implement the message-passing technique is to support heterogeneous computing environments. In particular, our approach focuses on taking full advantage of the hardware of each computing node, thus explicitly avoiding the bottlenecks introduced by the slowest computing node in

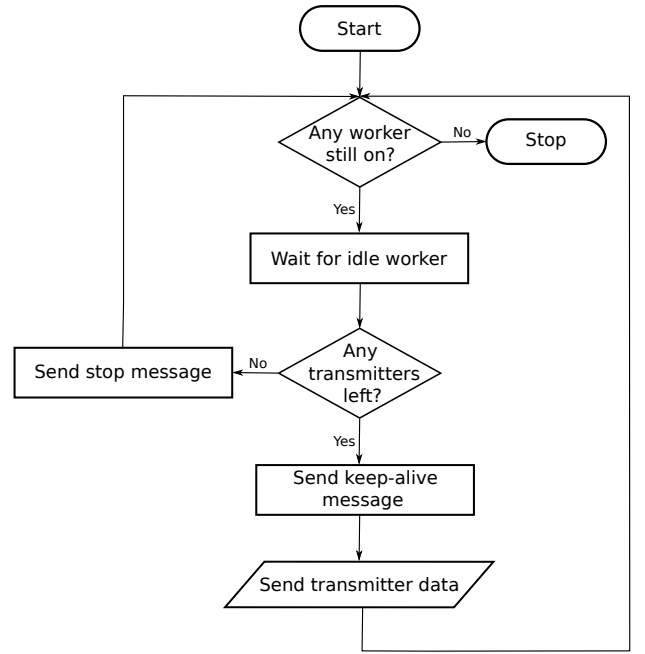


Figure 6. Flow diagram of the “Processing loop” step of the master process.

the cluster. This problem appears when evenly distributing the data among the worker processes on disparate hardware, as in [2], [18], being more noticeable with a larger number of computing nodes and processes. In other words, computing nodes that deliver better performance have more calculations assigned to them. Moreover, in real-world scenarios, it is often the case that a large number of dedicated computing nodes featuring exactly the same configuration is difficult to find, i.e., not every organization owns a computer cluster.

A second reason for selecting a message-passing technique is related to the flexibility it provides for load balancing, which is of greater importance when dealing with extra data or information besides just spatial data [16]. This can be seen in Figure 6 on page 7, where the master process, before delivering the spatial subset and transmitter-configuration data, sends a message to the worker process, indicating that it is about to receive more work. This a priori meaningless message plays a key role in correctly supporting the asynchronous process communication. Notice that the subset of spatial data that a worker process receives is directly related to the transmitter for which the prediction will be calculated. Similar to [38], [39], this problem-specific property enables the use of a data-decomposition technique based on a block partition of spatial data, e.g., the DEM and clutter data.

In general, there are many different ways a parallel program can be executed, because the steps from the different processes can be interleaved in various ways and a process can make non-deterministic choices [35], which may lead to situations such as race conditions [9] and deadlocks. A deadlock occurs whenever two or more running processes are waiting for each other to finish, and thus neither ever does. To prevent PRATO from deadlocking, message sending and receiving should be paired, i.e., an equal number of send and receive messages on

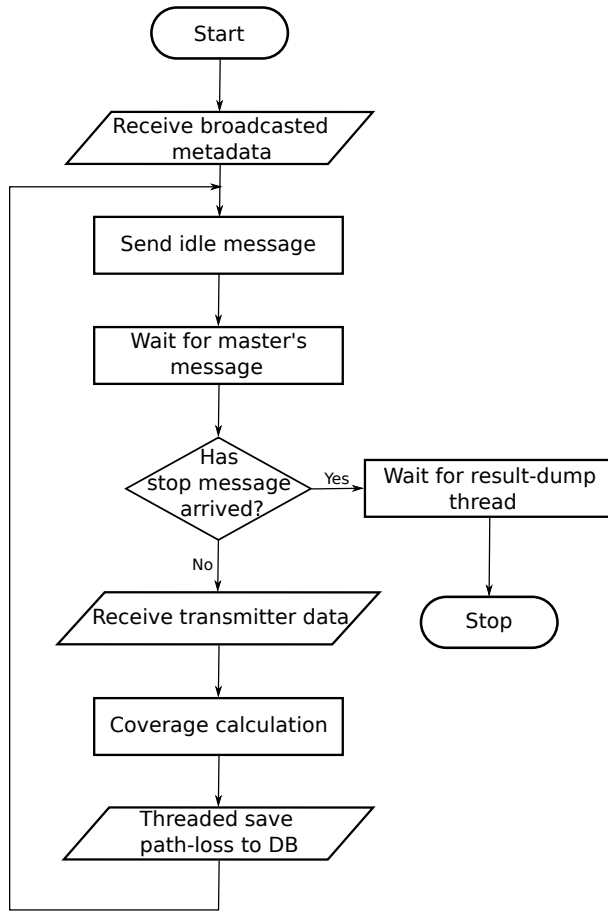


Figure 7. Flow diagram of a worker process.

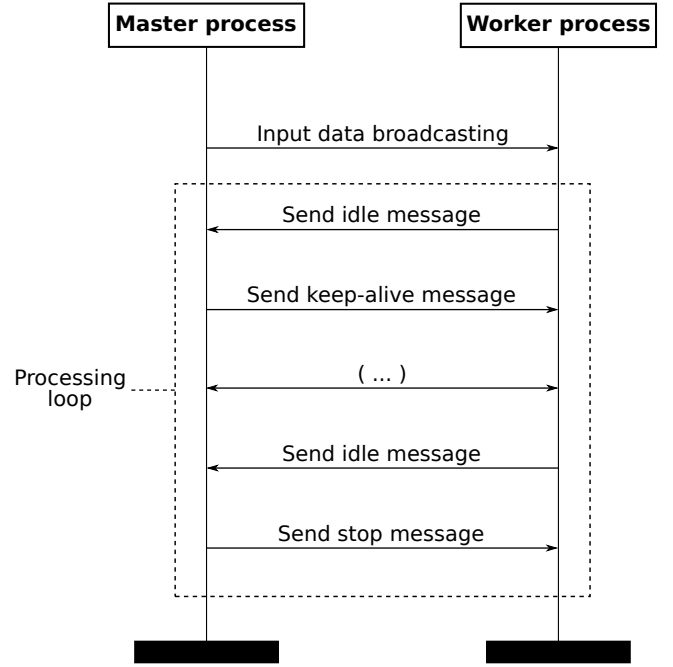


Figure 8. Communication diagram, showing message passing between master and one worker process.

the master and worker sides [35].

Figure 8 on page 8 depicts the master-worker message passing, from which the transmitter-data transmission has been excluded for clarity. Notice how each idle message sent from the worker process is paired with an answer from the master process, whether it is a keep-alive or a stop message.

V. SIMULATIONS

Considering the large computational power needed for predicting the radio-coverage of a real mobile network, the use of a computer cluster is recommended. A computer cluster is a group of interconnected computers that work together as a single system. Computer clusters typically consist of several commodity PCs connected through a high-speed local-area network (LAN) with a distributed file system, like NFS [34]. One such system is the DEGIMA cluster [14] at the Nagasaki Advanced Computing Center (NACC) of the Nagasaki University in Japan. This system ranked in the TOP 500 list of supercomputers until June 2012¹, and in June 2011 it held third place in the Green 500 list² as one of the most energy-efficient supercomputers in the world.

This section presents the simulations and analyses of the parallel version of PRATO. Our aim is to provide an exhaustive

analysis of the performance and scalability of the parallel implementation in order to achieve the objectives of this work. The most common usage case for PRATO is to perform a radio-coverage prediction for multiple transmitters. Therefore, a straight-forward parallel decomposition is to divide a given problem instance by transmitter, for which each coverage prediction is calculated by a separate worker process.

The following simulations were carried out on 34 computing nodes of the DEGIMA cluster. The computing nodes are connected by a LAN, over a Gigabit Ethernet interconnect. As mentioned before, the reason for using a high-end computer cluster such as DEGIMA is to explore by experimentation the advantages and drawbacks of the introduced methods. However, this does not imply any loss of generality when applying these principles over a different group of networked computers, i.e., not acting as a computer cluster.

Each computing node of DEGIMA features one of two possible configurations, namely:

- Intel Core i5-2500T quad-core processor CPU, clocked at 2.30 GHz, with 16 GB of RAM; and
- Intel Core i7-2600K quad-core processor CPU, clocked at 3.40 GHz, also with 16 GB of RAM.

During the simulation runs, the nodes equipped with the Intel i5 CPU host the worker processes, whereas the master process and the PostgreSQL database server (version 9.1.4) each run

¹<http://www.top500.org>

²<http://www.green500.org>

on a different computing node, featuring an Intel i7 CPU. The database server performs all its I/O operations on the local file system, which is mounted on an 8 GB RAM disk. During the simulations, the path-loss predictions of 5,120 transmitters occupied less than 4 GB of this partition.

All the nodes are equipped with a Linux 64-bit operating system (Fedora distribution). As the message passing implementation we use OpenMPI, version 1.6.1, which has been manually compiled with the distribution-supplied gcc compiler, version 4.4.4.

A. Test networks

To test the parallel performance of PRATO, we prepared different problem instances that emulate real radio networks of different sizes. In order to create the synthetic test data-sets with an arbitrary number of transmitters, we used the real data of a group of 2,000 transmitters, which we randomly replicate and distribute over the whole target area. The configuration parameters of these 2,000 transmitters were taken from the LTE network deployed in Slovenia by Telekom Slovenije, d.d. The path-loss predictions were calculated using the Walfisch-Ikegami model. The digital elevation model has an area of 20,270 km², with a resolution of 25 m². The clutter data extends over the same area and resolution, containing different levels of signal loss due to land usage. For all the points within a radius of 20 km around each transmitter, we assume that the receiver is positioned 1.5 m above the ground, and the frequency is set to 1,843 MHz.

B. Weak scalability

This set of simulations is meant to analyze the scalability of the parallel implementation in cases where the workload assigned to each process (one MPI process per processor core) remains constant as we increase the number of processor cores and the total size of the problem, i.e., the number of transmitters deployed over the target area is directly proportional to the number of processor cores and worker processes. We do this by assigning a constant number of transmitters per core, while increasing the number of cores hosting the worker processes. Here we test for the following numbers of transmitters per worker/core: {5, 10, 20, 40, 80}, by progressively doubling the number of worker processes from 1 to 64.

Problems that are particularly well-suited for parallel computing exhibit computational costs that are linearly dependent on the size of the problem. This property, also referred to as algorithmic scalability, means that proportionally increasing both the problem size and the number of cores results in a roughly constant time to solution.

The master-worker (MW) configuration performs result aggregation continuously, i.e., while receiving the intermediate results from the worker processes. In contrast, the master-worker-DB (MWD) setup performs the result aggregation as the final step. With this set of experiments, we would like to investigate how the proposed MWD technique compares with the classic MW approach in terms of scalability when dealing with different problem instances and numbers of cores.

An important fact about the presented simulations when using multi-threaded implementations is to avoid oversubscribing a computing node. For example, if deploying four worker processes over a quad-core CPU, the extra threads will have a counter effect on the parallel efficiency, since the CPU resources would be exhausted, which slows the whole process down. For this reason, we have deployed three worker processes per computing node, leaving one core free for executing the extra threads.

1) *Results and discussion:* The results represent the best running time out of a set of 20 independent simulation runs, in which we randomly selected both the transmitters and the rank ordering of the worker processes. The collected running times for the weak-scalability experiments are shown in Figure 9 on page 9. All the measurements express wall-clock times in seconds for each setup and problem instance, defined as the number of transmitters per process (TX/process). The wall-clock time represents the real time that elapses from the start of the master process to its end, including the time that passes while waiting for the resources to become available. The running-time improvements of the master-worker-DB against the master-worker setup are shown in Table II.

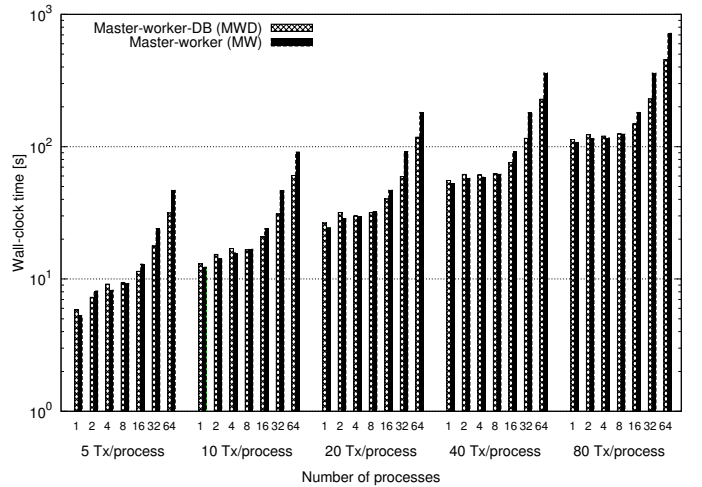


Figure 9. Measured wall-clock time for weak-scalability experiments, featuring MW and MWD setups. Experiments allocate one MPI worker process per core. The wall-clock time axis is expressed in a base-10 logarithmic scale, whereas the axis representing the number of cores is expressed in a base-2 logarithmic scale.

Table II
RUNNING-TIME GAIN (IN PERCENT) OF THE SIMULATIONS FOR THE WEAK-SCALABILITY OF THE MWD SETUP RELATIVE TO THE CLASSIC MW APPROACH.

TX/core	Number of cores						
	1	2	4	8	16	32	64
5	-11.39	-10.42	-11.14	-0.95	11.75	26.15	32.53
10	-5.84	-7.78	-7.67	0.91	12.81	33.28	33.55
20	-8.59	-10.88	-1.04	1.95	14.29	35.23	35.27
40	-5.26	-6.90	-3.68	-0.67	17.27	36.23	36.65
80	-5.29	-7.11	-3.20	-0.31	17.94	36.32	36.57

The time measurements observed from the weak-scalability results show that the classic MW approach performs well

for up to four worker processes. When using eight worker processes, the MW setup is practically equivalent to the MWD approach, indicating that the master process is being fully exploited. When increasing the problem size and the number of worker processes to 16, the running-time gain is already clear, favoring the MWD configuration. This gain keeps growing, although slower, as we increase the number of worker processes to 32 and 64, confirming our hypothesis that in a classic MW approach, the parallel efficiency is bounded by the capacity of the master process to serve an increasing number of worker processes. Interestingly, the gain when using 32 and 64 worker processes is almost the same. After further investigation, we found the reason for this behavior was due to the LAN being completely saturated by the worker processes. Consequently, they have to wait for the network resources to become available before sending or receiving data, which is not the case when running the MW setup. Therefore, using the MWD approach we hit a hardware constraint, meaning that the bottleneck is no longer at the implementation level. Moreover, since the master process is far from overloaded when serving 64 worker processes, we can expect the MWD approach will keep scaling if we use a faster network infrastructure, e.g., 10-gigabit Ethernet or InfiniBand.

Certainly, the parallel version of PRATO, when using the MWD approach, scales better when challenged with a large number of transmitters (5,120 for the biggest instance) over 64 cores. This fact shows PRATO would be able to calculate the radio-coverage prediction for real networks in a feasible amount of time, since many operational radio networks have already deployed a comparable number of transmitters, e.g., the 3G network within the Greater London Authority area, in the UK [26].

Not being able to achieve perfect weak scalability using the MWD setup is due to a number of factors. Specifically, the overhead time of the serial sections of the parallel process grow proportionally with the number of cores, e.g., aggregation of the intermediate results, although the total contribution of this overhead remains low for large problem sizes. Moreover, the communication overhead grows linearly with the number of cores used. Consequently, we can confirm the findings of [18], who concluded that the data-set size should be large enough for the communication overhead to be hidden by the calculation time, for parallelization to be profitable in terms of a running-time reduction.

C. Strong scalability

This set of simulations is meant to analyze the impact of increasing the number of computing cores for a given problem size, i.e., the number of transmitters deployed over the target area does not change, while only the number of worker processes used is increased. Here we test for the following number of transmitters: {1,280, 2,560, 5,120}, by gradually doubling the number of workers from 1 to 64 for each problem size.

1) *Results and discussion:* Similar to the weak-scalability experiments, these time measurements show that when applying a classic MW approach the running-time reduction

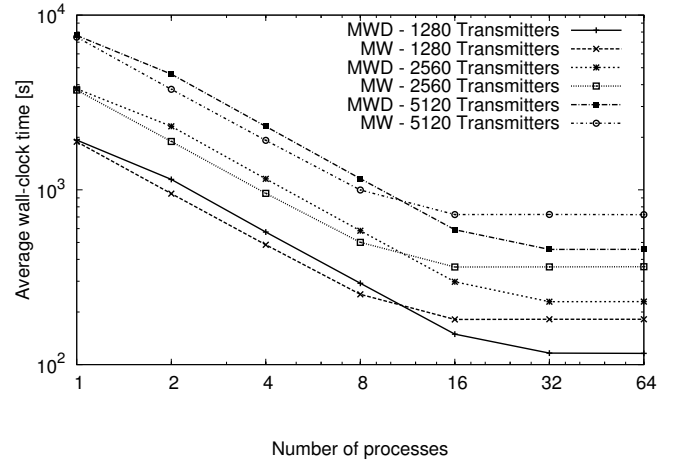


Figure 10. Measured wall-clock time for strong-scalability experiments, featuring MW and MWD setups. Experiments assigned one MPI worker process per core. The wall-clock time axis is expressed in a base-10 logarithmic scale, whereas the axis representing the number of cores is expressed in a base-2 logarithmic scale.

starts flattening when more than eight worker processes are used. Moreover, the running times for 16, 32 and 64 worker processes are the same, i.e., it does not improve due to the master process being saturated. In contrast, when using our MWD technique, the running-time reduction improves for up to 32 worker processes, after which there is no further improvement since the network is being fully exploited. These results clearly show that when applying parallelization using a larger number of worker processes, the master process becomes the bottleneck of the MW approach. When using the MWD configuration, a steady running-time reduction is observed, until a hardware constraint is hit, e.g., the network infrastructure.

We have also measured the overhead of sending/receiving asynchronous messages in order to support heterogeneous systems, which is lower than 0.02% of the total running time for the MW experiments, and 0.01% for the MWD experimental set.

In order to further analyze how well the application scales using the MW and MWD approaches, we measured the performance of the parallel implementation in terms of its speedup, which is defined as

$$S(NP) = \frac{\text{execution time for base case}}{\text{execution time for } NP \text{ cores}}, \quad (6)$$

where NP is the number of cores executing the worker processes. As the base case for comparisons we chose the parallel implementation running on only one core and decided against using the serial implementation. We consider that the serial implementation is not a good base comparison for the parallel results as it does not reuse the resources between each transmitter-coverage calculation and it does not overlap the I/O operations with the transmitter computations. In practice, this means that several concatenated runs of the serial version would be considerably slower than the parallel but single worker implementation.

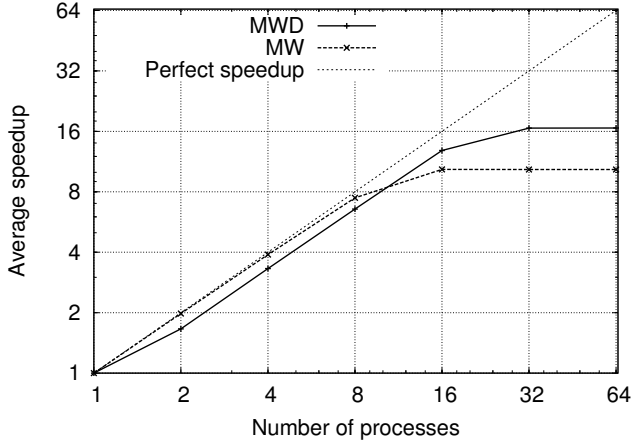


Figure 11. Average speedup for strong-scalability experiments. The speedup axis is expressed in a base-2 logarithmic scale, and the axis representing the number of cores is expressed in a base-2 logarithmic scale.

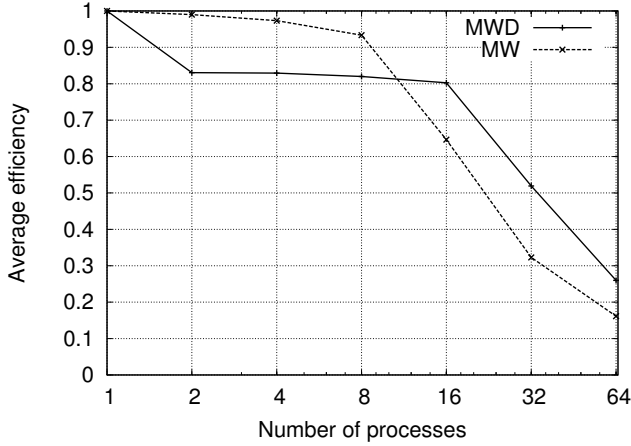


Figure 12. Average parallel efficiency for strong-scalability experiments. The parallel-efficiency axis is expressed in a linear scale, whereas the axis representing the number of cores is expressed in a base-2 logarithmic scale.

Using the speedup metric, linear scaling is achieved when the obtained speedup is equal to the total number of processors used. However, it should be noted that a perfect speedup is almost never achieved, due to the existence of serial stages within an algorithm and the communication overhead of the parallel implementation.

Figure 11 on page 11 shows the average speedup of the parallel implementation for up to 64 worker processes, using the standard MW method and our MWD approach. The average speedup was calculated for the three different problem instances, i.e., 1,280, 2,560, and 5,120 transmitters deployed over the target area. The number of transmitters used in these problem sizes is comparable to several real-world radio networks that were already deployed in England, e.g., Hampshire County with 227 base stations, West Midlands with 414 base stations, and Greater London Authority with 1,086 base stations [26]. Note that it is common for a single base station to host multiple transmitters.

The plotted average speedup clearly shows the minimal overhead of the MWD approach when using a small number of worker processes. This overhead accounts for the final

aggregation of the intermediate results at the DB, which in the MW configuration is performed along worker processing. Like before, the DB component allows the parallel implementation to fully exploit the available computing resources when deploying a larger number of worker processes, until the network-speed limit is met. Of course, these results are directly correlated with the wall-clock times shown in Figure 10 on page 10.

Another measure to study how well PRATO utilizes the available computing resources considers the parallel efficiency of the implementation, i.e., how well the parallel implementation makes use of the available processor cores. The definition of parallel efficiency is as follows

$$E(NP) = \frac{S(NP)}{NP}, \quad (7)$$

where $S(NP)$ is the speedup as defined in Equation (6), and NP is the number of cores executing worker processes. Figure 12 on page 11 shows the average parallel efficiency of the parallel implementation for different problem sizes as we increase the number of processing cores. Like for the speedup measure, we have calculated the average parallel efficiency from the three problem instances analyzed.

The ideal case for a parallel application would be to utilize all the available computing resources, in which case the parallel efficiency would always be equal to one as we increase the core count. From the plot in Figure 12 on page 11, we can see that the efficiency of the MWD approach is better than in the MW case for larger number of processes and as long as there is still capacity at the LAN level. In accordance to the previous analysis, the under utilization of the computing resources is more significant when the master process is overloaded (in the MW case) than when the network infrastructure is saturated (in the MWD case). The lower efficiency is directly proportional to the number of idle worker processes that are waiting for the master process (MW case) or for network access (MWD case).

Overall, the experimental results confirm that the objective of fully exploiting the available hardware resources is accomplished when applying our MWD approach, thus improving the scalability and efficiency of PRATO when compared with a traditional MW method.

VI. CONCLUSION

We have presented PRATO, a parallel radio-coverage prediction tool for radio networks. The tool, as well as the patterns for exploiting the computing power of a group of networked computers, i.e., a computer cluster, are intended to be used for spatial analysis and decision support. The introduced MWD technique, which combines the use of a database system with a work-pool approach, delivers improved performance when compared with a traditional MW setup. Moreover, the presented system provides parallel and asynchronous computation, that is completely independent of the GIS used, in this case the GRASS environment. Consequently, a GIS installation is needed on only one of the nodes, thus simplifying the required system setup and greatly enhancing the applicability of this methodology in different environments.

The extensive simulations, performed on the DEGIMA cluster of the Nagasaki Advanced Computing Center, were analyzed to determine the level of scalability of the implementation, as well as the impact of the presented methods for parallel-algorithm design aimed at spatial-data processing. The conducted analyses show that when using the MWD approach, PRATO is able to calculate the radio-coverage prediction of real-world mobile networks in a feasible amount of time, which is not possible for a serial implementation. Moreover, the experimental results show PRATO has a better scalability than the standard MW approach, since it is able to completely saturate the network infrastructure of the cluster. These promising results also show the great potential of our MWD approach for parallelizing different time-consuming spatial problems, where databases form an intrinsic part of almost all GIS. Furthermore, the automatic optimization of radio networks, where millions of radio-propagation predictions take part in the evaluation step of the optimization process, are also excellent candidates for this approach. Indeed, this last point is currently undergoing extensive research and it is already giving its first results.

Encouraged by the favorable results, further work will include abstracting the introduced MWD principle into a multi-purpose parallel framework such as Charm++ [20], which provides a functionality for overlapping execution and communication, as well as fault tolerance.

In addition, as PRATO is also a free and open-source software project³, it can be readily modified and extended to support, for example, other propagation models and post-processing algorithms. This characteristic provides it with a clear advantage when compared to commercial and closed-source tools.

ACKNOWLEDGMENTS

This project was co-financed by the European Union, through the European Social Fund. Hamada acknowledges support from the Japan Society for the Promotion of Science (JSPS) through its Funding Program for World-leading Innovative R&D on Science and Technology (First Program).

REFERENCES

- [1] S Akhter, K Aida, and Y Chemin. Grass gis on high performance computing with mpi, openmp and ninf-g programming framework. In *Proceeding of ISPRS 2010*, 2010.
- [2] S. Akhter, Y. Chemin, and K. Aida. Porting a GRASS raster module to distributed computing Examples for MPI and Ninf-G. *OSGeo Journal*, 2(1), 2007.
- [3] Marc P Armstrong, Mary Kathryn Cowles, and Shaowen Wang. Using a computational grid for geographic information analysis: a reconnaissance. *The Professional Geographer*, 57(3):365–375, 2005.
- [4] R. Blazek and L. Nardelli. The GRASS server. In *Proceedings of the Free/Libre and Open Source Software for Geoinformatics: GIS-GRASS Users Conference*, 2004.
- [5] D.R. Butenhof. *Programming with POSIX threads*. Addison-Wesley Professional, 1997.
- [6] I. Campos, I. Coterillo, J. Marco, A. Monteoliva, and C. Oldani. Modelling of a Watershed: A Distributed Parallel Application in a Grid Framework. *Computing and Informatics*, 27(2):285–296, 2012.
- [7] D.J. Cichon and T. Kurner. Propagation prediction models. *COST 231 Final Rep*, 1995.
- [8] Andrea Clematis, Mike Mineter, and Richard Marciano. Guest editorial: high performance computing with geographical data. *Parallel Computing*, 29(10):1275–1279, 2003.
- [9] C. Clemencon, J. Fritscher, M. Meehan, and R. Rühl. An implementation of race detection and deterministic replay with MPI. *EURO-PAR’95 Parallel Processing*, pages 155–166, 1995.
- [10] Leila De Floriani, Paola Magillo, and Enrico Puppo. Applications of computational geometry to geographic information systems. *Handbook of computational geometry*, pages 333–388, 1999.
- [11] Zhaoya Gong, Wenwu Tang, David A Bennett, and Jean-Claude Thill. Parallel agent-based simulation of individual-level spatial interactions within a multicore computing environment. *International Journal of Geographical Information Science*, 27(6):1152–1170, 2013.
- [12] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: portable parallel programming with the message passing interface*, volume 1. MIT press, 1999.
- [13] Qingfeng Guan, Phaeton C Kyriakidis, and Michael F Goodchild. A parallel computing approach to fast geostatistical areal interpolation. *International Journal of Geographical Information Science*, 25(8):1241–1267, 2011.
- [14] T. Hamada and K. Nitadori. 190 TFlops astrophysical N-body simulation on a cluster of GPUs. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–9. IEEE Computer Society, 2010.
- [15] Masaharu Hata. Empirical formula for propagation loss in land mobile radio services. *Vehicular Technology, IEEE Transactions on*, 29(3):317–325, 1980.
- [16] Kenneth A Hawick, Paul David Coddington, and HA James. Distributed frameworks and parallel algorithms for processing large-scale geographic data. *Parallel Computing*, 29(10):1297–1333, 2003.
- [17] A. Hrovat, I. Ozimek, A. Vilhar, T. Celcer, I. Saje, and T. Javornik. Radio coverage calculations of terrestrial wireless networks using an open-source GRASS system. *WSEAS Transactions on Communications*, 9(10):646–657, 2010.
- [18] F. Huang, D. Liu, X. Tan, J. Wang, Y. Chen, and B. He. Explorations of the implementation of a parallel IDW interpolation algorithm in a Linux cluster-based parallel GIS. *Computers & Geosciences*, 37(4):426–434, 2011.
- [19] Qunying Huang, Chaowei Yang, Karl Benedict, Abdelmounaam Rezgui, Jibo Xie, Jizhe Xia, and Songqing Chen. Using adaptively coupled models and high-performance computing for enabling the computability of dust storm forecasting. *International Journal of Geographical Information Science*, 27(4), 2013.
- [20] Laxmikant V. Kale and Abhinav Bhatele, editors. *Parallel Science and Engineering Applications: The Charm++ Approach*. Taylor & Francis Group, CRC Press, November 2013.
- [21] Xia Li, Xiaohu Zhang, Anthony Yeh, and Xiaoping Liu. Parallel cellular automata for large-scale urban simulation using load-balancing techniques. *International Journal of Geographical Information Science*, 24(6):803–820, 2010.
- [22] Christian Mehlführer, Josep Colom Colom Ikuno, Michal Šimko, Stefan Schwarz, Martin Wrulich, and Markus Rupp. The Vienna LTE simulators-Enabling reproducibility in wireless communications research. *EURASIP Journal on Advances in Signal Processing*, 2011(1):1–14, 2011.
- [23] Gordon E Moore et al. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.
- [24] A. Neskovic and N. Neskovic. Microcell electric field strength prediction model based upon artificial neural networks. *AEU-International Journal of Electronics and Communications*, 64(8):733–738, 2010.
- [25] Markus Neteler and Helena Mitasova. *Open Source software and GIS*. Springer, 2008.
- [26] Ofcom. Table of base station totals. Available from: <http://stakeholders.ofcom.org.uk/sitefinder/table-of-totals/>, 2012.
- [27] A. Osterman. Implementation of the r.cuda.los module in the open source GRASS GIS by using parallel computation on the NVIDIA CUDA graphic cards. *Elektrotehniški Vestnik*, 79(1-2):19–24, 2012.
- [28] M Tamer Özsu and Patrick Valduriez. *Principles of distributed database systems*. Springer, 2011.
- [29] Giuseppe Piro, Luigi Alfredo Grieco, Gennaro Boggia, Francesco Capozzi, and Pietro Camarda. Simulating LTE cellular systems: an open-source framework. *Vehicular Technology, IEEE Transactions on*, 60(2):498–513, 2011.
- [30] Rolf Rabenseifner, Georg Hager, and Gabriele Jost. Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. In *Parallel, Distributed and Network-based Processing*, 2009

³The source code is available for download from <http://cs.ijs.si/benedicic/>

- 17th Euromicro International Conference on, pages 427–436. IEEE, 2009.
- [31] A.B. Saleh, S. Redana, J. Hämäläinen, and B. Raaf. On the coverage extension and capacity enhancement of inband relay deployments in LTE-Advanced networks. *Journal of Electrical and Computer Engineering*, 2010:4, 2010.
 - [32] T.K. Sarkar, Z. Ji, K. Kim, A. Medouri, and M. Salazar-Palma. A survey of various propagation models for mobile communication. *Antennas and Propagation Magazine, IEEE*, 45(3):51–82, 2003.
 - [33] N. Shabbir, M.T. Sadiq, H. Kashif, and R. Ullah. Comparison of Radio Propagation Models for Long Term Evolution (LTE) Network. *arXiv preprint arXiv:1110.1519*, 2011.
 - [34] S. Shepler, M. Eisler, D. Robinson, B. Callaghan, R. Thurlow, D. Noveck, and C. Beame. Network file system (NFS) version 4 protocol. *Network*, 2003.
 - [35] S. Siegel and G. Avrunin. Verification of halting properties for MPI programs using nonblocking operations. *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 326–334, 2007.
 - [36] Iana Siomina and Di Yuan. Minimum pilot power for service coverage in WCDMA networks. *Wireless Networks*, 14(3):393–402, June 2007.
 - [37] Michael Stonebraker. SQL databases v. NoSQL databases. *Communications of the ACM*, 53(4):10–11, 2010.
 - [38] Siham Tabik, Luis Felipe Romero, and Emilio López Zapata. High-performance three-horizon composition algorithm for large-scale terrains. *International Journal of Geographical Information Science*, 25(4):541–555, 2011.
 - [39] Siham Tabik, A Villegas, and Emilio López Zapata. Optimal tilt and orientation maps: a multi-algorithm approach for heterogeneous multicore-GPU systems. *The Journal of Supercomputing*, pages 1–13, 2013.
 - [40] Siham Tabik, Emilio L Zapata, and Luis F Romero. Simultaneous computation of total viewshed on large high resolution grids. *International Journal of Geographical Information Science*, 27(4):804–814, 2013.
 - [41] A. Valcarce, G. De La Roche, Á. Jüttner, D. López-Pérez, and J. Zhang. Applying FDTD to the coverage prediction of WiMAX femto-cells. *EURASIP Journal on Wireless Communications and Networking*, 2009:1–13, 2009.
 - [42] Mladen A Vouk. Cloud computing—issues, research and implementations. *Journal of Computing and Information Technology*, 16(4):235–246, 2008.
 - [43] Shaowen Wang. A cyberGIS framework for the synthesis of cyber-infrastructure, GIS, and spatial analysis. *Annals of the Association of American Geographers*, 100(3):535–557, 2010.
 - [44] Michael J Widener, Neal C Crago, and Jared Aldstadt. Developing a parallel computational implementation of amoeba. *International Journal of Geographical Information Science*, 26(9):1707–1723, 2012.
 - [45] Howard Xia, Henry L Bertoni, Leandro R Maciel, Andrew Lindsay-Stewart, and Robert Rowe. Radio propagation characteristics for line-of-sight microcellular and personal communications. *Antennas and Propagation, IEEE Transactions on*, 41(10):1439–1447, 1993.
 - [46] Ling Yin, Shih-Lung Shaw, Dali Wang, Eric A Carr, Michael W Berry, Louis J Gross, and E Jane Comiskey. A framework of integrating GIS and parallel computing for spatial control problems—a case study of wildfire control. *International Journal of Geographical Information Science*, 26(4):621–641, 2012.